Gap Processing for Adaptive Maximal Poisson-Disk Sampling

Dong-Ming Yan* KAUST

Peter Wonka† KAUST

Abstract

In this paper, we study the generation of maximal Poisson-disk sets with varying radii. First, we present a geometric analysis of gaps in such disk sets. This analysis is the basis for maximal and adaptive sampling in Euclidean space and on manifolds. Second, we propose efficient algorithms and data structures to detect gaps and update gaps when disks are inserted, deleted, moved, or have their radius changed. We build on the concepts of the regular triangulation and the power diagram. Third, we will show how our analysis can make a contribution to the state-of-the-art in surface remeshing.

CR Categories: I.3.6 [Computer Graphics]: Methodology and Techniques—Poisson-disk Sampling;

Keywords: Adaptive Poisson-disk sampling, maximal sampling, gaps, regular triangulation, power diagram, blue noise, remeshing

1 Introduction

We study the geometry of gaps in disk sets. Given a set of disks with varying radii in a compact domain Ω in Euclidean space or on a manifold, we would like to know if they fully cover the domain or if there is a gap, i.e. uncovered space to insert other disks into the disk set. We are interested in knowing if gaps exist, where the gaps are, and how to implement efficient gap processing operations.

There are three important papers that conduct such an analysis in the context of 2D Poisson-disk sampling: Dunbar and Humphreys [2006], Jones [2006], and Ebeida et al. [2011b]. In the first part of this paper, we will present a simpler and more natural analysis that additionally extends to 1) disk sets with varying radii, 2) arbitrary dimensions, and 3) 2-manifolds. The main idea of our approach is to study gaps in the context of the regular triangulation and the power diagram [Aurenhammer 1991] of the point set of the disk centers.

Besides the general curiosity about an interesting geometric problem, gaps in disk sets play an important role in sampling applications, e.g., sample generation for ray tracing, image stippling, video stippling, environment map sampling, surface remeshing, plant ecosystem simulation, texture synthesis, video synthesis, and particle-based simulation. By analyzing this set of interesting applications to find commonalities, we identified several operations that need to be performed efficiently: gap detection, gap clustering, gap primitive extraction, updating gap primitives when points are deleted or inserted, updating gap primitives when points move, and updating gap primitives when the sampling radius changes. The second part of the paper will introduce algorithms for these operations, based on the analysis in the first part.

While our algorithms will not improve all aforementioned applications, in the third part of this paper we investigate an application to surface remeshing (as well as 2D meshing), where we can successfully improve the state-of-the-art in aspects such as minimal angle, vertex valence and triangle quality. We will discuss why remeshing benefits from blue-noise properties, maximal sampling, bounds on vertex valence, and geometric bounds (e.g. angle bounds). We propose a remeshing framework that jointly optimizes these criteria and evaluate it in a comparison to several recent remeshing algorithms. The main contributions of these three parts are:

- A simple and elegant theoretical analysis of the gap geometry that improves the work of [Ebeida et al. 2011b] for 2D, and is the first analysis for d-D and manifolds.
- The design of efficient algorithms and data structures for all the gap processing operations identified above.
- A surface remeshing (as well as 2D meshing) algorithm that compares favorably to the state-of-the-art in aspects such as minimal angle, vertex valence and triangle quality.

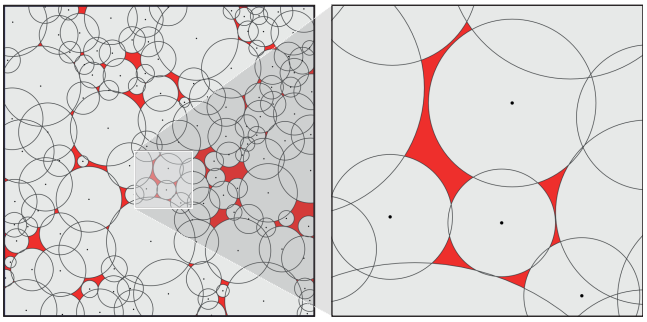


Figure 1: *Left: Illustration of gaps of a disk set; the uncovered regions are shown in red. Right: zoom in view of the uncovered regions.*

1.1 Related work

We briefly review techniques for generating random point sets, but refer the reader to [Lagae and Dutré 2008] for a more comprehensive survey. Our review focuses on the treatment of gaps in these algorithms.

There are a large number of sampling algorithms that produce point sets with different attributes [Lloyd 1982; Dippé and Wold 1985; Cook 1986; Mitchell 1987; Mitchell 1991; McCool and Fiume 1992; Turk 1993] that are either not concerned with the geometry of gaps or do not use an acceleration data structure. Our work can improve upon some of these methods, e.g. [McCool and Fiume 1992].

Poisson-disk sampling: Most recent algorithms for efficient Poisson-disk sampling maintain a data structure of gap primitives. The simplest primitive is a square, because it is easy to sample [White et al. 2007; Bridson 2007; Gamito and Maddock 2009; Ebeida et al. 2012] and it can be efficiently subdivided. These algorithms are extremely efficient when trying to fill a large mainly uncovered region, but even when filling small gaps, they are efficient enough so that the overall running time is much faster than the algorithm built on an exact representation of gaps [Dunbar and Humphreys 2006]. Ebeida et al. [2011b] propose a hybrid approach that first uses squares and later convex polygons bounding the intersections of a square and multiple circles as gap primitives. Jones [2006] uses a Voronoi diagram to extract gap primitives. In

*e-mail: yandongming@gmail.com

†e-mail: pwonka@gmail.com

contrast, we use a power diagram and the dual regular triangulation for our analysis, which results in a more general and much simpler solution.

Adaptive sampling: Fattal presents an adaptive sampling algorithm based on kernel density estimation [Fattal 2011]. Kalantari et al. [2011] propose to use joint distributions by breaking down the 2D probability density function (pdf) into a 1D conditional pdf (cpdf) and a 1D pdf for adaptive sampling. However, the proposed algorithm is dependent on a threshold for discretizing the 1D pdf, which cannot achieve maximal sampling. In a recent work, Mitchell et al. [2012] study the 2D Poisson-disk sampling with various radii. However, the maximality is not discussed.

Sampling on surfaces: Fu and Zhou [2008] generalize the scalloped sector based sampling [Dunbar and Humphreys 2006] to 3D mesh surfaces, and present an isotropic remeshing algorithm by extracting a mesh from the samples. Lloyd iterations are used to further smooth the resulting mesh. Cline et al. [2009] propose dart throwing algorithms on surfaces based on a hierarchical triangulation. Bowers et al. [2010] extend the parallel sampling method [Wei 2008] to mesh surfaces and introduce a spectrum analysis for uniform surface sampling algorithms as well. Wei and Wang [2011] present a framework for spectral analysis of nonuniform blue noise sampling for both 2D domains and surfaces. Corsini et al. [2012] present an algorithm for surface blue noise sampling based on a space subdivision combined with a pre-generation of the samples. Xu et al. [2012] extend the concept of CCVT [Balzer et al. 2009] by introducing capacity constrained Delaunay triangulations for blue noise sampling on surfaces. Chen et al. [2012] combine the Centroidal Voronoi Tessellation (CVT) [Yan et al. 2009] and the capacity area constraint [Balzer et al. 2009] for blue noise sampling on surfaces. Again, none of these above mentioned methods satisfy the maximal sampling property. Although many surface blue noise sampling algorithms have been proposed recently, few of them use this blue noise sampling for geometric applications, such as remeshing.

Other approaches: There are several extensions to Poisson-disk sampling that we did not consider in our current framework, but that might be interesting avenues for future work, e.g. parallel sampling [Wei 2008; Gamito and Maddock 2009] and multi-class sampling [Wei 2010]. Other applications that pre-compute tile sets that can then be quickly arranged in real-time [Ostromoukhov et al. 2004; Kopf et al. 2006] might benefit from our work in the preprocessing phase.

2 Theoretical Gap Analysis

We study a set of disks $\mathcal{D} = \{(\mathbf{p}_i, r_i)\}_{i=1}^n$ in a compact d-D domain Ω, where $\mathbf{p}_i$ and r_i are the center and the radius of the i^{th} disk, respectively. Here we use the notion *disk* to represent the general disk, i.e., sphere in 3D, and hyper-sphere in d-D ($d > 3$). We assume that the center $\mathbf{p}_i$ of any disk cannot be covered by the other disks, i.e., $\forall i, j\,(i \neq j), \|\mathbf{p}_i - \mathbf{p}_j\| \geq \max(r_i, r_j)$. If we draw a disk at each center $\mathbf{p}_i$ with radius r_i, the domain Ω will be fully or partially covered. We are interested in the properties of the uncovered region, which is defined as $\Omega - \mathcal{D}$. As shown in Fig. 1, if the domain Ω is partially covered by a set of disks, the uncovered region is split into a set of isolated regions. We use the term *gap* to refer to a connected component of the uncovered region.

At a first glance, a gap can be arbitrarily complex or can be arbitrarily small so that it is very difficult to sample gaps directly. Therefore, like most previous work we shall decompose these complex gaps to a set of gap primitives GP_i that satisfy 1) the union of these covers the gaps $\Omega - \mathcal{D} \subseteq \cup GP_i$, 2) the gap primitives are non-intersecting $\forall i, j\,(i \neq j), GP_i \cap GP_j = \emptyset$, and 3) it is easy to sample a gap primitive using rejection sampling.

2.1 Power diagram and regular triangulation

We analyze the geometric properties of gaps based on the concept of the power diagram, and its dual, called regular triangulation. The disk set $\mathcal{D}$ is represented by a set of weighted points $\mathcal{P}_w = \{(\mathbf{p}_i, w_i)\}_{i=1}^n$, where $w_i = r_i^2$. The power of two weighted points is defined as

$$\Pi(\mathbf{p}_i, w_i, \mathbf{p}_j, w_j) = \|\mathbf{p}_i - \mathbf{p}_j\|^2 - w_i - w_j.$$

Then the power diagram $\mathcal{PD}$ of $\mathcal{P}_w$ is a set of non-overlapping power cells $\{\Omega_i\}_{i=1}^n$, such that

$$\Omega_i = \{\mathbf{x} \in \Omega \mid \Pi(\mathbf{p}_i, w_i, \mathbf{x}, 0) \leq \Pi(\mathbf{p}_j, w_j, \mathbf{x}, 0), \forall j \neq i\},$$

A vertex of the power diagram is called a *power vertex* and an edge is called a *power edge*. Note that a d-D power diagram can be interpreted as the intersection of a $(d+1)$-D Voronoi diagram and a d-D hyperplane [Ash and Bolker 1986].

The regular triangulation of $\mathcal{P}_w$ is a d-D simplicial complex $\mathcal{T}$ and it is the dual of the power diagram. A d-D simplicial complex is the union of a set of k-simplices ($0 \leq k \leq d$). Each k-simplex is the convex hull of $k+1$ linear independent points, that are dual to a $(d-k)$-D convex polyhedron of the power diagram. The simplicial complex $\mathcal{T}$ satisfies: i) any face of a simplex is also in $\mathcal{T}$, and ii) the intersection of any two simplices is either empty, or is a face of both simplices. We denote the set of the d-simplices (i.e., triangles in 2D and tetrahedra in 3D) as $\mathcal{RT} = \{t_j\}$. For each $t \in \mathcal{RT}$, there exists a point $\mathbf{c}_t \in \Omega$ that has equal powers w.r.t. the $d+1$ vertices of t, and this power is less than the power of $\mathbf{c}_t$ w.r.t. any other weighted points in the triangulation. Each $\mathbf{c}_t$ is called the *power center* of t, which is a power vertex of $\mathcal{PD}$. We denote the power of a d-simplex t by $\Pi(t) = \Pi(\mathbf{p}, w, \mathbf{c}_t, 0)$, where $\mathbf{p}$ refers to one of the vertices of t, and w is the corresponding weight of $\mathbf{p}$. The power diagram and the regular triangulation are equivalent to the Voronoi diagram and the Delaunay triangulation when the weights of all the points are the same.

2.2 Existence of gaps

In the following, we will give the theorems that state the condition of the existence of gaps in d-D spaces, and on 3D surfaces.

Theorem 2.1 *A gap exists iff* $\exists t \in \mathcal{RT}, \Pi(t) > 0$.

Here we will give the proof only in 2D (see Fig. 2), as it is more intuitive and the proof for higher dimensions is similar.

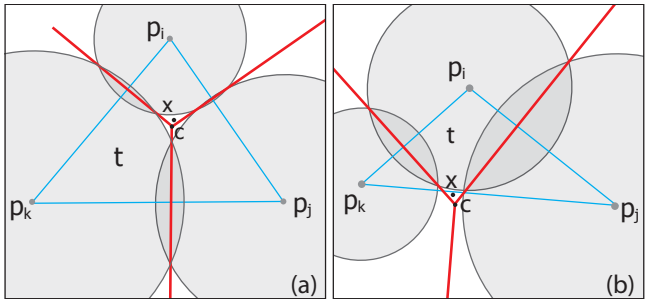


Figure 2: *Existence of gaps in the 2D plane. The power center* $\mathbf{c}$ *of a triangle* $\triangle \mathbf{p}_i\,\mathbf{p}_j\,\mathbf{p}_k$ *can be used to test for the existence of gaps.*

$\Rightarrow$ Suppose that there is a gap. For any point $\mathbf{x}$ in this gap, we define the function $F(\mathbf{x})$ as the minimal power between the point $\mathbf{x}$ and the

weighted points $\mathcal{P}_w$. Suppose that $\mathbf{x}$ falls in the power cell Ω_i of the sample $(\mathbf{p}_i, w_i)$, clearly, $F(\mathbf{x}) = \Pi(\mathbf{x}, \mathbf{p}_i)$ by definition of the power diagram. $F(\mathbf{x})$ attains its maxima when $\mathbf{x}$ coincides with a power vertex $\mathbf{c}$, which is the power center of a triangle $t \in \mathcal{RT}$. When $\mathbf{x}$ is in the gap, by definition $F(\mathbf{x}) > 0$ since $\Pi(\mathbf{x}, \mathbf{p}_i) > 0$. Note that $\Pi(t) = \Pi(\mathbf{c}, \mathbf{p}_i) = \max(F(\mathbf{x}))$. Hence $\Pi(t) > F(\mathbf{x}) > 0$.

$\Leftarrow$ If $\exists t$, $\Pi(t) > 0$, then power center of the triangle t is not covered. So there exists a gap. □

Gaps on 3D surfaces: If the input domain Ω is a 3D surface, Thm. 2.1 does not apply any more since the domain is not flat. In this case, we generalize the approach of Jones [2006] to 3D surfaces using the concept of the *Restricted Power Diagram* ($\mathcal{RPD}$).

The restricted power diagram is a generalization of the *Restricted Voronoi Diagram* (RVD) [Edelsbrunner and Shah 1997] on surfaces. The $\mathcal{RPD}$ is defined as the intersection of the 3D power diagram and the surface, i.e., $\mathcal{RPD}(\mathcal{P}_w) = \mathcal{PD}(\mathcal{P}_w) \bigcap \Omega = \{\Omega_i \bigcap \Omega\}_{i=1}^{n}$.

Theorem 2.2 *A gap of the disk set $\mathcal{D}$ (sphere set in 3D) on a 3D surface exists iff there exists a sample $\mathbf{p}_i$ whose corresponding restricted power cell is not fully covered by the sphere $(\mathbf{p}_i, r_i)$ centered at $\mathbf{p}_i$.*

Thm. 2.2 describes the global condition of gap existence on surfaces. The proof is straightforward and we omit it here. In Sec. 5.3, we will show how to locally detect gaps on mesh surfaces in the context of ϵ-sampling and *Restricted Regular Triangulations* ($\mathcal{RRT}$).

3 2D Gap Primitive Processing

In the following, we describe the most important operations related to gap processing in the 2D plane. We provide a short overview of each of these operations here and in the subsections we will describe each operation in more detail:

- Gap detection: gap detection refers to analyzing an existing point set to determine if gaps exist and where they are. If no gap exists, we can determine that a point set is maximal. In our solution, this operation outputs a list of *gap triangles* (see Sec. 3.1.
- Gap clustering: this operation groups all gap triangles belonging to the same independent gap set (Sec. 3.2) together. In our proposed solution, this clustering is performed on the level of gap triangles, before the actual gaps are extracted. This step is optional but it enables a simple parallelization of gap sampling.
- Gap primitive extraction: given a set of input points, this operation computes a set of gap primitives covering all gaps (Sec. 3.3).
- Gap primitive updates: these operations update the gap primitives after points are inserted, deleted, moved, or when one or multiple disk radii are changed (Sec. 3.4).

The state of the framework is the weighted point set $\mathcal{P}_w$ and the corresponding regular triangulation $\mathcal{RT}$. At the beginning, $\mathcal{RT}$ has to be initialized once.

3.1 Gap detection

We can optionally collect all the gap triangles in an array for later use or simply return a boolean to indicate the existence of gaps. Note that a gap triangle does not necessarily contain an uncovered region itself. See Fig. 4(b) for an example: the triangle t_0 is fully covered but $\Pi(t_0) > 0$. The incident gap region is outside of the triangle t_0. We traverse all the triangles in $\mathcal{RT}$, compute the power center c_i and power $\Pi(t_i)$ of each triangle t_i. If $\Pi(t_i) > 0$, triangle t_i is marked as a *gap triangle* according to Thm. 2.1.

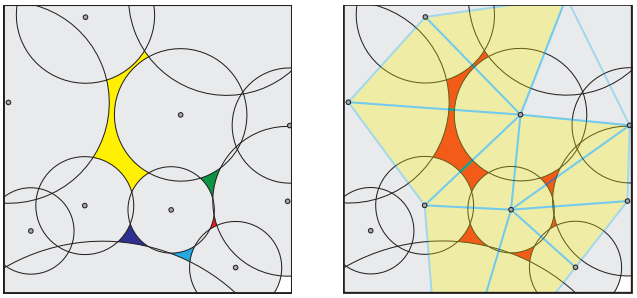


Figure 3: *Gap clustering. There are five gaps (left) that clustered together forming an independent gap set (right) by the clustering algorithm.*

3.2 Gap clustering

We define the *independent gap set* (IGS) as a collection of gaps, where 'independent' means that if we put a new disk in a gap of an IGS, it will not affect the gap geometry of gaps in other IGSs. Fig. 3 shows such an example. We also use IGS to represent the collection of incident gap triangles or primitives with slight abuse of notation.

We can group gap triangles that are incident to the same IGS. In our implementation, the algorithm is performed for gap triangles, but there is a simple one to one correspondence to gap primitives (see Sec. 3.3). All the gap triangles are marked as unvisited at the beginning. We then traverse all the gap triangles. Each time when we encounter an unvisited triangle, we extract the IGS using a simple region growing algorithm, i.e., if two neighboring triangles are gap triangles, then they belong to the same group of an IGS. The grouped triangles are marked as visited and the traversal continues. This step ends when all the gap triangles are marked as visited. There are three cases of the relationship between two neighboring gap triangles, which are called *connectivity rules.* To cluster IGSs, all three rules apply, but if only connected components are desired, the last rule does not apply.

Connectivity rules: For a given pair of neighboring gap triangles t_0, t_1, and their power centers $\mathbf{c}_0, \mathbf{c}_1$, respectively (Fig. 4), we classify as follows:

- The length of the common edge shared by t_0, t_1 is larger than $r_0 + r_1$, where r_0, r_1 are the corresponding sampling radii at the vertices of the common edge $\mathbf{p}_0, \mathbf{p}_1$ (Fig. 4(a) and 4(b)).
- The length of the common edge shared by t_0, t_1 is smaller than $r_0 + r_1$, but $\mathbf{c}_0, \mathbf{c}_1$ are on the same side of the common edge (Fig. 4(c)).
- The length of the common edge shared by t_0, t_1 is smaller than $r_0 + r_1$, and $\mathbf{c}_0, \mathbf{c}_1$ are on different sides of the common edge (Fig. 4(d)). In this case, although two gaps are disconnected, putting a new disk in one empty region may affect the geometry on the other side. So the two triangles are classified as belonging to the same IGS.

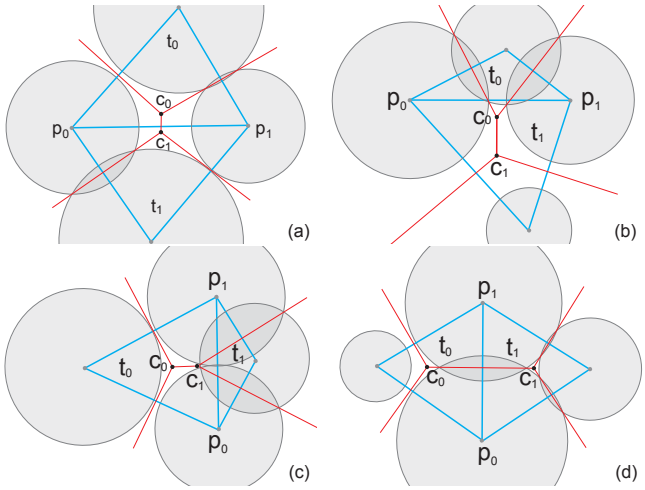


Figure 4: *Illustration of the connectivity between two neighboring gap triangles t_0, t_1 (blue triangles), and c_0, c_1 are the corresponding power centers, respectively.* $\mathbf{p}_0\,\mathbf{p}_1$ *are on the common edge shared by* t_0, t_1. *(a)* $|\mathbf{p}_0\,\mathbf{p}_1| > r_0 + r_1$. c_0, c_1 *on different sides of* $\mathbf{p}_0\,\mathbf{p}_1$. *(b)* $|\mathbf{p}_0\,\mathbf{p}_1| > r_0 + r_1$, c_0, c_1 *on the same side of* $\mathbf{p}_0\,\mathbf{p}_1$. *(c)* $|\mathbf{p}_0\,\mathbf{p}_1| < r_0 + r_1$, c_0, c_1 *are on the same side of* $\mathbf{p}_0\,\mathbf{p}_1$. *(d)* $|\mathbf{p}_0\,\mathbf{p}_1| < r_0 + r_1$ *but* c_0, c_1 *are on different sides of* $\mathbf{p}_0\,\mathbf{p}_1$.

3.3 Gap primitive extraction

The algorithm extracts a *gap primitive* for each triangle of the gap separately. Each gap primitive is a simple convex polygon with up to six edges. The *connectivity rules* are used to assist the decomposition. Given a triangle t_0 belonging to a gap, we simply traverse the three neighbors of t_0 and extract the vertices of the gap primitive based on the classification of the types of neighboring triangles. Assuming that t_1 is one of t_0's neighbors, and $\mathbf{c}_0, \mathbf{c}_1$ are power centers of t_0 and t_1, respectively. $\mathbf{p}_0\,\mathbf{p}_1$ is the common edge shared by t_0 and t_1, and the direction of $\mathbf{p}_0\,\mathbf{p}_1$ is *ccw* (counter-clockwise) in triangle t_0. Then there can be the following two cases:

First, if the triangle t_0 contains only the power center $\mathbf{c}_0$ of itself (see Fig. 5(a)-(d)), then we can extract the gap primitive directly by traversing its three (ccw) edges using the following rules:

- If $|\mathbf{p}_0\,\mathbf{p}_1| \leq r_0 + r_1$, we compute the intersection points of the disks centered at $\mathbf{p}_0$ and $\mathbf{p}_1$, the intersection point on the left side of $\mathbf{p}_0\,\mathbf{p}_1$ is added to the gap primitive of t_0.
- If $|\mathbf{p}_0\,\mathbf{p}_1| > r_0 + r_1$, we compute the intersection points of the disks and edge $\mathbf{p}_0\,\mathbf{p}_1$, the resulting points $\mathbf{p}_{01}$ and $\mathbf{p}_{10}$ are added to the gap primitive of t_0.

Second, the triangle t_0's power center is outside t_0, e.g., lies in its neighbor triangle t_1. Note that in complex cases, t_0's power center can also lie in its neighbor's neighbor triangle. All these triangles belong to the same connected gap component, according to the connectivity rules. This case is equivalent to that of a triangle containing more than one power center. To extract gap primitives is equivalent to decompose the gap w.r.t. each gap triangle. We handle this configuration as follows:

- For the edges we currently traversed , if the power centers of two incident triangles lie on two different sides, we compute the intersection points as before.
- For the edges with two power centers of two incident triangles laying on the same side, e.g., edge $\mathbf{p}_0\,\mathbf{p}_1$ shown in Fig. 5(e), we simply connect t_0's power center $\mathbf{c}_0$ and $\mathbf{p}_0$, $\mathbf{p}_1$, which results in two intersection points A and B. The segment AB is then added to the primitive polygon of t_0. Similarly, if the neighboring triangle's power center lies on the same side of of the common edge of the current triangle, e.g., t_1 in Fig. 5(e), we also add AB to t_1's gap primitive.

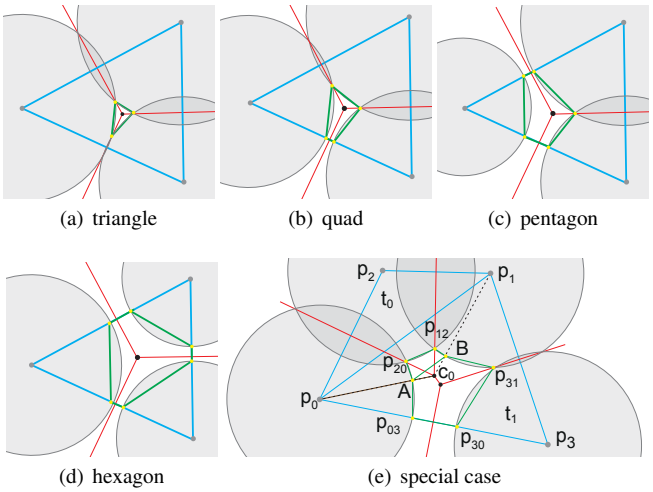


Figure 5: *(a)-(d) Simple gap primitives contained inside the triangle, and (e) a special case when the power center is outside of the triangle.*

The above extraction process decomposes a connected gap component into non-overlapping simple polygons. We provide more details in Appendix A to show the validness of this process.

Boundary handling: We also consider bounded domains. The boundary can be a simple polygon with or without holes. In these cases, we first compute the clipped power diagram using the technique presented in [Yan et al. 2012]. The gaps that touch the domain boundary are simply computed by clipping the power cells that contain parts of gaps with the surrounding disks. Fig. 9 shows a result of meshing a 2D polygon.

3.4 Gap primitive updates

Appropriate algorithms exist to insert and delete points from a regular triangulation. Deleting k points requires $O(k)$ time. Inserting points in the worst case requires $O(\log(n))$, or $O(\sqrt{n})$ time if the points are uniform randomly distributed, where n is the total number of points. However, in our case the insertion is always done through dart throwing in a gap primitive so that it is a completely local operation that can be performed in constant time.

If all points move in a coherent fashion, the power diagram can be updated efficiently using existing algorithms [de Castro et al. 2009], but in our implementation we simply rebuild the data structures if all points move at once. Movement of individual points is handled by deletion and insertion. Changing the sampling radius of all disks causes a change in the combinatorial structure and we also have to rebuild the regular triangulation.

4 Gap Computation on Surfaces

In this section, we generalize the gap analysis and computation framework to mesh surfaces. Suppose that the input domain $\Omega = \{f_j\}_{j=1}^{m}$ is a triangle mesh surface (where f_j refers to a triangle) and

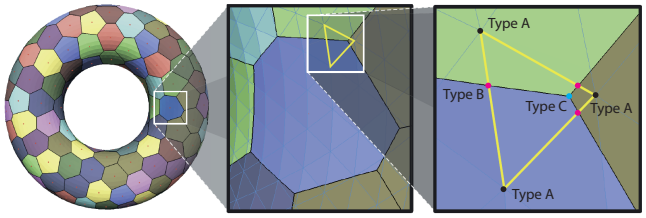


Figure 6: *Illustration of the restricted power diagram on a surface (left); middle: a restricted power cell; right: a mesh triangle is split into polygons shared by the incident cells.*

$\mathcal{P}_w$ is a set of weighted samples lying on the surface. Recall that in Sec. 2.2 we have defined the restricted power diagram on surfaces. In this case, each restricted power cell is the intersection of the 3D power cell and the mesh surface, i.e., $\Omega_i \bigcap \Omega = \{\bigcup\{\Omega_i \bigcap f_j\}, \forall f_j \in \Omega\}$. Fig. 6 shows an example of the $\mathcal{RPD}$ of a torus. There are three types of vertices in the $\mathcal{RPD}$ of a mesh surface, called restricted power vertices, i.e., A) the original vertices of the mesh, B) the intersection of a mesh edge and a bi-sector power plane, and C) the intersection of a power edge and a mesh triangle (Fig. 6(right)). Thm. 2.2 can be adapted as follow:

Theorem 4.1 *A gap of a disk set $\mathcal{P}_w$ exists on a mesh surface iff there exists a restricted power cell of a sample $\mathbf{p}_i$, whose restricted power vertices are not fully covered by the respective sphere $(\mathbf{p}_i, r_i)$ centered at $\mathbf{p}_i$ with radius r_i.*

In the above definition, a type C restricted power vertex is also called a *restricted power center*. Note that each restricted power center is dual to a triangle of the so called restricted regular triangulation ($\mathcal{RRT}$). The triangles of the $\mathcal{RRT}$ are not lying on the surface, only the vertices do. Since the restricted triangles are no longer lying on the surface, we cannot use the 2D per-triangle gap computation (Sec. 3) for meshes. Alternatively, based on Thm. 4.1, we present an approach similar to [Jones 2006] to compute the gap primitives on surfaces. The gap primitives are computed by clipping the restricted power cell by the sphere centered at each weighted point. Each restricted power cell can be split into a set of triangles. Then the clipping problem is reduced to a triangle-sphere intersection problem. The clipped regions, i.e., gap primitives, are approximated by a set of triangles and associated with their incident gaps, as shown in Fig. 7. The details of the $\mathcal{RPD}$ computation are analogue to the restricted Voronoi diagram computation in [Yan et al. 2009].

Now we are able to identify and compute gaps on mesh surfaces. However, in the maximal sampling framework (see Sec. 5.1), we are willing to cluster the independent gap sets as in the 2D counterpart, which can be used for parallel gap filling. A simple region-growing based approach can be used by detecting the connectivity between clipped gap primitives. Furthermore, if the initial sampling $\mathcal{P}_w$ adapts to the local properties of the mesh surface (e.g., *ϵ-sampling property* [Amenta and Bern 1999]) and the *topological ball property*p [Edelsbrunner and Shah 1997] is met, then the dual restricted regular triangulation is homeomorphic to the input domain Ω. In this case we are able to define the 3D gap triangles similar to the 2D plane. A gap triangle is a triangle of the $\mathcal{RRT}$ with at least one vertex whose restricted power cell is not fully covered by the sphere centered at the vertex. Once the gap triangles are detected, we group the neighboring gap triangles/primitives into IGSs for further processing.

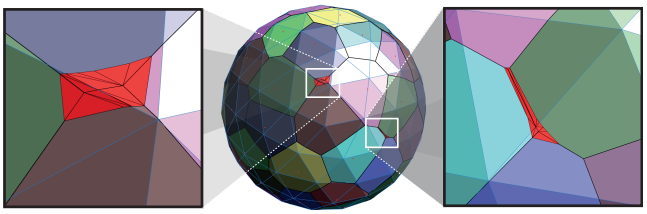


Figure 7: *Gap computation on surfaces by triangle-sphere clipping. Gaps are shown in red.*

5 Surface sampling and remeshing

In this section, we first describe a framework for adaptive Poisson-disk sampling on surfaces, which also works well in 2D. Then we give a brief discussion of the link between Poisson-disk sets and surface remeshing. Moveover, we present a high quality surface remeshing algorithm, as well as a randomized mesh optimization algorithm built on top of the sampling framework, which greatly improves the sampling/meshing quality.

5.1 Adaptive sampling on surfaces

As input we use a mesh $\Omega = \{f_i\}_{i=1}^m$, a minimal sampling radius r_{min}, a maximal sampling radius r_{max} (default value $16\,r_{min}$), as well as a density function $\rho(\mathbf{x})$ defined on the mesh surface. A voxel grid is built for accelerating the sampling process. We first voxelize the mesh with voxels whose sizes are equal to $\frac{r_{min}}{\sqrt{3}}$. Each voxel records the indices of samples that fully cover it. A voxel is *valid* if it is not fully covered by any sphere. We follow the two steps sampling strategy presented in [Ebeida et al. 2011b]: 1) dart throwing with a grid and 2) gap filling.

Initial sampling: In the first step, we perform classic dart throwing on surfaces. Each triangle f is associated with a weight $w_f = \rho(\mathbf{c}_f)|f|$, where $\mathbf{c}_f$ is the barycenter and $|f|$ is the area of the triangle. The cumulative probability density function (cpdf) of the weights is stored in a flat array. Each time a dart is generated by first selecting a triangle from the cpdf, then randomly generating a point $\mathbf{p}$ in the triangle, as well as a radius $r = \min(r_{max}, \max(r_{min}, \frac{1}{\sqrt{\rho(\mathbf{p})}}))$ associated with the point. The new dart is tested against the grid. The dart is accepted if it is not contained by previous samples and does not contain any other existing samples. The index of the new sample is recorded by the voxels that are fully contained by this sample. The first step is terminated when k consecutive rejections are observed ($k = 300$ in our implementation).

Gap filling: The second step is an iterative algorithm. At each iteration, all gap primitives are triangulated and each triangle is associated with a weight as before. For each independent gap set, we create a cpdf for the triangles belonging to this set. Since the independent gap sets will not affect each other, we perform gap filling in parallel. Note that in the gap filling step, the newly generated disks may cover existing samples. In this case, we re-compute the radius of the new disk so that it will not contain any previous samples, i.e., we set the radius of the new disk as the distance to the nearest sample. Here we choose the maximal conflict metric [Kalantari and Sen 2011] since we made the assumption that the center of disks cannot be covered by any other disks. This solution is reasonable for a smooth density function. While there are multiple applications for sampling points on surfaces, the most important application is remeshing which we will describe next.

5.2 Poisson-disk sampling and surface remeshing

Surface remeshing is a broad topic and the best choice of a surface remeshing algorithm depends on the application [Alliez et al. 2008]. Poisson-disk sampling is mainly useful for remeshing for simulation. In other applications, e.g. architectural panel layouts, the blue noise pattern will often be considered unattractive. Before presenting our solution to adaptive remeshing, we would like to first argue what criteria make a good mesh and also establish the link between our disk sampling and remeshing. This will explain why our work can be more successful in remeshing for simulation than previous surface sampling algorithms.

First, the blue noise property is important because it reduces directional bias in the simulation [Ebeida et al. 2012]. While this fact has been used for uniform remeshing using disk sampling with a global uniform radius, it is also important for adaptive remeshing. Second, the most important geometric quality characteristic of a triangle mesh is the minimal angle in a triangle or the percentage of triangles with small angles [Shewchuk 2002]. Statistics relating to the minimal angle, as well as $Q(t) = \frac{6}{\sqrt{3}} \frac{|t|}{p(t)\,h(t)}$, (where $|t|$ is the area of t, $p(t)$ is the half-perimeter of t and $h(t)$ the longest edge length of t [Frey and Borouchaki 1997]) are present in almost all recent remeshing papers. The reason for this is that the minimal angle influences the condition number of the matrices in FEM. Interestingly, there is a theoretical guarantee that the minimal angle is 30 degrees in a uniform 2D maximal sampling [Chew 1989; Ebeida et al. 2011a]. It can also be shown that the same conclusion still holds on surfaces. This takes out the main point we had before. That a smooth density function gives some guarantees. We will propose a randomized optimization technique which can improve the angle bounds dramatically for adaptive remeshing (see Sec. 5.3). This is an important link that is often overlooked in previous work: maximal sampling is essential for theoretical guarantees about triangle angles. Maximal sampling is also important in praxis. We verified this by generating 100 point sets with $10k$ samples and then removing the last 20 sampled points. The minimal angle is significantly worse. Only in about 10% of the cases it is still ≥ 30. Third, irregular vertices are undesirable, because they require special handling and often separate code. Therefore, we aim at reducing the different types of irregular vertices to only two types: valence 5 and valence 7 vertices. It would also be helpful to reduce the total number of irregular vertices, but this is conflicting with the blue noise property and the minimal angle properties. Fourth, the surface approximation is important. Most commonly, this is measured in the Hausdorff distance or the root mean squared error. Fifth, it is important to use Euclidean distances and not geodesic distances. While sampling geodesic disks seems more complex and sophisticated, this can be counterproductive for remeshing. The edges in a mesh are straight and not geodesic path on surfaces. A simple implication is that the use of geodesic distances voids all angle guarantees.

5.3 Surface remeshing and optimization

To obtain a mesh, we extract the dual triangulation from the restricted power diagram of the samples, which is a good remeshing of the input surface Ω. There are several potential challenges to surface remeshing that we address here.

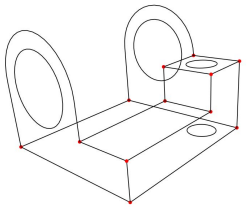
Topological validity: In this work we adapt the sampling radius to the local feature size to ensure the topological validity. Theoretically, the ϵ-sampling theorem [Amenta et al. 1998] requires that for each point $\mathbf{x} \in \Omega$, there exists a sample $\mathbf{p}$, such that $|\mathbf{x}-\mathbf{p}| < 0.3\,lfs(\mathbf{p})$, then the $\mathcal{RRT}$ of the samples is homeomorphic to Ω. However, in practice, even these theoretical guarantees are not met, our algorithm still generates topological correct results as shown in Fig. 15. We can detect topological inconsistencies using the algorithm presented in [Yan et al. 2009]. Once such a configuration is detected, we discard the samples that cause the problem and re-sample the gaps. We refer to [Yan et al. 2009] for more details of the mesh extraction algorithm since it is not the main contribution of this paper.

Valence and angle optimization: The uniform blue noise remeshing has many nice properties. Given a constant sampling radius r, the meshes generated from blue noise sampling exhibit the following bounds: the angle bound $[30^o, 120^o]$, the edge length bound $[r, 2\,r]$ and the area bound $[\frac{\sqrt{3}}{4}\,r^2, \frac{3\sqrt{3}}{4}\,r^2]$ as its 2D counterpart. All these properties are desired in many applications, especially the angle bound is crucial for FEM applications [Shewchuk 2002]. However, in the case of the adaptive sampling, the above theoretical bounds do not hold any more. To improve the meshing quality, we introduce a novel and simple randomized optimization algorithm, i.e., angle bound optimization and valence optimization.

The optimization iteratively removes the sample points with unsatisfactory properties and their neighborhoods and then re-fills the gaps. In angle bound optimization, the vertices with one triangle angle less than a minimal angle threshold or larger than a maximal threshold are removed. In valence optimization, the vertices whose degrees are less than 5 or larger than 7 are removed. These two optimization criteria can either be performed separately, or jointly. The valence/angle optimization terminates when the required criteria are met or the maximal iteration number (25 in our implementation) is reached. In a joint valence and angle optimization, a global optimization is performed interleaving between valence and angle optimization. Typically it takes 5-10 global iterations to meet both quality requirements. During the optimization, the $\mathcal{RRT}$ and $\mathcal{RPD}$ are locally updated.

Edge length optimization: Besides the valence/angle optimization, we are also able to optimize other geometric properties, such as edge length, using the same framework. Given a user input threshold λ_e, we iteratively remove all the edges $(\mathbf{p}_i, \mathbf{p}_j)$ with $|\mathbf{p}_j - \mathbf{p}_i| > \lambda_e\,(r_i + r_j)$, followed by a gap filling step, until the desired edge length quality is met.

Feature preservation: We also implemented a simple feature preserving sampling step in our framework. The features are provided by the user, in the form of a 1D curved skeleton (see figure below). The corners (vertices of the skeleton with more than 2 neighboring edges, or with sharp turning angles) are inserted directly as sample points, and the feature curves are first sampled before the surface sampling. The edge length of the feature samples are optimized by edge length optimization ($\lambda_e = \sqrt{3}$) so that the neighboring spheres are deeply intersected [Cheng et al. 2007]. The samples of the feature skeleton remain fixed during the surface sampling/optimization stages.



6 Results and Discussion

In this section, we present several sampling and meshing results of our framework based on the proposed gap processing techniques. We also compare our results with current state-of-the-art approaches. We use CGAL [cga] for the sequential 2D/3D regular triangulation, and the OpenMP library for parallel gap filling. The experimental results are conducted on an Intel X5680 Dual Core 3.33GHz CPU with 4GB memory and a 64-bit Windows 7 operating system.

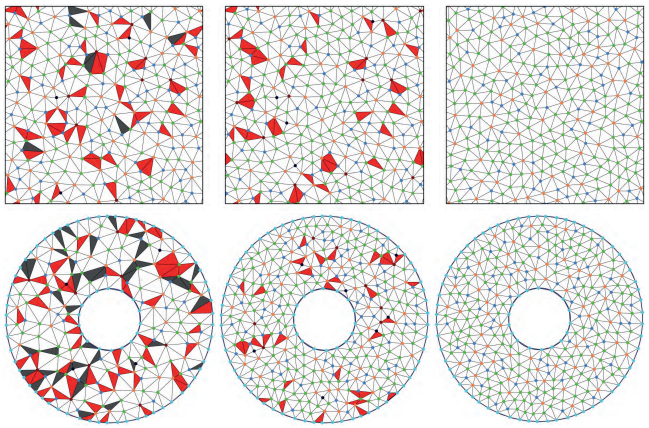


Figure 8: *Illustration of uniform 2D sampling/optimization. Top: a unit square with periodic boundary condition; bottom: two concentric circles. Left column: dart throwing; middle: maximal sampling; right: optimized sampling (angle and valence). Vertices with valence 5: blue, 6: green, 7: orange. Darker points correspond to higher(> 7) or lower(< 5) valences. The triangles with $\theta_{min} < 30^o$ or $\theta_{max} > 120^o$ are shown in dark gray, triangles with $\theta_{min} \in [30^o, 35^o]$ or $\theta_{max} \in [105^o, 120^o]$ are shown in red.*

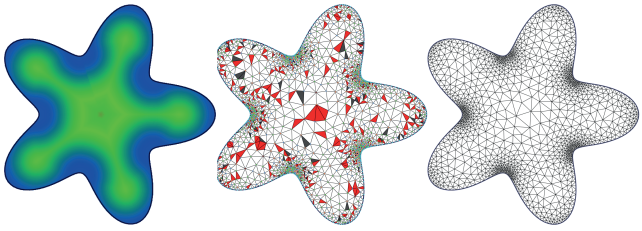


Figure 9: *Adaptive 2D sampling/meshing. Left: density field; middle: initial sampling. Triangles with $\theta_{min} < 30^o$ and $\theta_{max} > 120^o$ are shown in dark gray and triangles with $\theta_{min} \in [30^o, 35^o]$ and $\theta_{max} \in [105^o, 120^o]$ are shown in red; right: optimized sampling.*

6.1 2D sampling and meshing

We first present the results of 2D sampling. Fig. 8 shows two examples of uniform sampling and optimization. The first example is a 2D square with periodic boundary condition, and the second is defined by two concentric circles. The statistics of these results are given in Table 1. The spectral analysis of the optimized sampling in the unit square is given in Fig. 10, which shows that our optimization framework preserves the blue noise properties well (see supplemental material for the comparison with previous approaches).

An example of adaptive meshing is given in Fig. 9, where we set the desired angle bounds to $[35^o, 105^o]$. For the examples with boundaries, we first perform a 1D maximal sampling on the boundary, then we apply edge length optimization to ensure that the boundary disks are deeply intersected [Cheng et al. 2007] (we set $\lambda_e = \sqrt{3}$ in all our experiments). The boundary samples remain fixed during the later sampling/optimization inside the boundary.

If the sampling radius is a constant r, then the regular triangulation/power diagram are equivalent to the Delaunay triangulation/Voronoi diagram. In this case, we can use the more efficient implementation of the Delaunay triangulation instead of the regular triangulation. More comparisons of 2D uniform Poisson-disk sampling are given in supplemental materials.

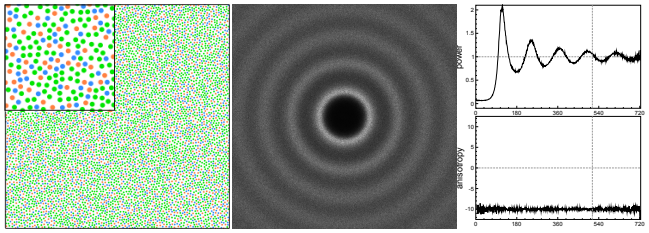


Figure 10: *Spectral analysis of the optimized sampling (10k samples) in the plane with angle bounds of $[35^o, 105^o]$ and only vertices of valence 5, 6, 7.*

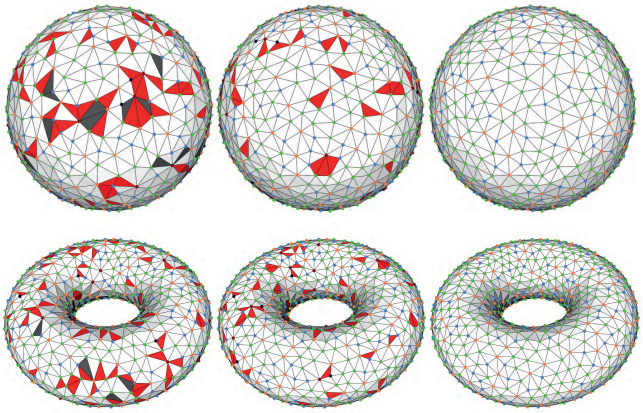


Figure 11: *Illustration of uniform sampling/optimization on surfaces (we set $r_{min} = 0.03$ for better visualization). Left column: dart throwing; middle: maximal sampling; right: optimized sampling (angle and valence). (Please refer to Fig. 9 for the color of vertices and triangles.)*

6.2 Surface sampling and remeshing

Uniform sampling/remeshing: We show several experimental results of surface remeshing and optimization. Fig. 11 shows the results of uniform sampling on a sphere and a torus. The minimal radius is set to $r_{pmin} = 4.5 \times 10^{-3}$. In this example, we generate three meshes for each model using: non-maximal sampling (dart throwing without gap filling), maximal sampling, and optimized sampling. The statistics of the meshes are shown in Table 1. We can see that the theoretical bounds do not hold for a non-maximal sampling (first experiment), while these bounds are guaranteed by maximal sampling (second experiment). For optimized (uniform) sampling, the desired angle bound is set to $[35^o, 105^o]$. This third experiment shows that the geometric properties are greatly improved.

We compare the uniform surface sampling with the most recent paper [Corsini et al. 2012]. As shown in Fig. 13(a), we are able to detect the gaps form their output and show that this competing result is not maximal. In another comparison of uniform sampling, the Elk model is remeshed with $31k$ vertices. We set the desired angle bound to $[37^o, 98^o]$ and still obtain a higher quality output, compared with the state-of-the-art [Yan et al. 2009]. The results are summarized in Table 2.

Adaptive remeshing: We applied our adaptive remeshing/optimization to various models. We use the *local feature size* (lfs) [Amenta et al. 1998] as density function, i.e., $\rho(\mathbf{x}) = \frac{1}{lfs(\mathbf{x})^2}$. We compare our remeshing algorithm with the state-of-the-art remeshing approaches [Cheng et al. 2007; Valette et al. 2008; Fu and Zhou 2008; Yan et al. 2009], in terms of the min quality, min/max angle,

Model	#v	θ_{min}	θ_{max}	$\|e\|'_{min}$	$\|e\|'_{max}$	$\|t\|'_{min}$	$\|t\|'_{max}$	$\theta < 30^o$	v_{567}
2D sampling/meshing									
Square1	26.8k	21.6	130.3	1.0	1.458	0.947	2.043	3.33	94.2%
Square2	34.5k	30.1	118.6	1.0	0.999	1.001	0.998	0.00	96.4%
Square3	35.1k	35.0	104.9	1.0	0.999	1.001	0.997	0.00	100%
Circles1	19.6k	18.9	131.4	1.0	1.950	1.088	2.950	8.83	93.1%
Circles2	24.9k	30.0	118.5	1.0	0.999	1.003	0.996	0.00	96.2%
Circles3	25.3k	35.0	105.0	1.0	0.999	1.003	0.995	0.00	100%
surface sampling/remeshing									
Sphere1	23.5k	22.1	129.3	1.0	1.618	0.993	2.182	1.12	94.5
Sphere2	29.7k	30.2	119.0	1.0	0.998	1.002	0.991	0.00	96.6
Sphere3	30.4k	35.0	105.0	1.0	0.998	1.002	0.996	0.00	100
Torus1	22.4k	21.3	126.8	1.0	1.382	0.979	1.816	0.88	94.6
Torus2	25.1k	30.1	119.1	1.0	0.998	1.004	0.994	0.00	96.3
Torus3	27.4k	35.1	104.9	1.0	0.999	1.003	0.995	0.00	100

Table 1: *Statistics of uniform sampling/meshing ($r_{min} = 4.5 \times 10^{-3}$). $|e|'_{min} = \frac{|e|_{min}}{r_{min}}$ is the ratio of minimal edge length over the theoretical minimal edge length bound, same for $|e|'_{max} = \frac{|e|_{max}}{2\,r_{min}}$, $|t|'_{min} = \frac{|t|_{min}}{\frac{\sqrt{3}}{4} r_{min}^2}$, and $|t|'_{max} = \frac{|t|_{max}}{\frac{3\sqrt{3}}{4} r_{min}^2}$. $\theta < 30^o$ is the percentage of the triangles with θ_{min} smaller than 30 degrees. v_{567} is the percent of vertices with valence 5, 6, or 7. For each model, we show the meshing quality of 1) classic dart throwing, 2) maximal sampling and 3) optimized sampling. The data shown in the table are averages of 10 runs for each model.*

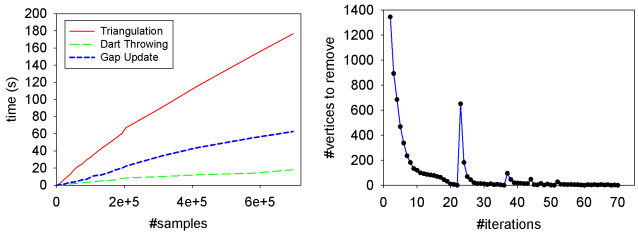


Figure 12: *Left: Timing curves of our sampling algorithm on the Bunny model. Right: convergence curve of the valence/angle optimization of Fig. 14. Each peak of the curve corresponds to a switch between valence/angle optimization.*

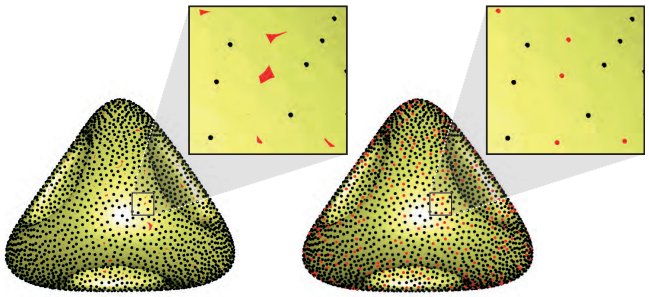


Figure 13: *Comparison with [Corsini et al. 2012]. Left: uniform sampling result of [Corsini et al. 2012] with 3909 samples ($r \approx 0.012$), gaps are detected by our technique (red regions); right: maximal sampling (4560 samples) by filling the gaps using our approach, and the new sampled points are shown in red.*

Hausdorff and RMS distances, the ratio of angles smaller than 30^o and the ratio of the valence 5,6,7 vertices (see Table. 2). In adaptive remeshing, the desired angle bound is set to $[32^o, 115^o]$. Similar to the uniform meshing case, we can observe that our approach exhibits better Q_{min} and θ_{min}, as well as high approximation quality to the input surface. Our method has larger Hausdorff distance compared with CVT-based remeshing due to the fact that we interpolate the input mesh while CVT tends to approximate the input mesh by minimizing the Hausdorff distance [Nivoliers et al. 2011].

We show an example of boundary handling in Fig. 16. The boundary is treated the same as a 1D feature curve. We compare our feature preserving remeshing result with previous work in Fig. 17. In this example, we request a minimal angle of 35^o, but we can only obtain 33.9^o. We also cannot get a pure valence $5, 6, 7$ solution because of the feature constraints.

Efficiency: Fig. 12(a) shows the timing curve of our algorithm with an increasing number of sample points, and the convergence of the valence/angle optimization is shown in Fig. 12(b) on the Bunny model shown in Fig. 14. For this example, our algorithm takes $12.4s$ for the initial sampling, $6.4s$ for gap filling and $4.9s$ for optimization ($23.7s$ in total), while CVT takes $182s$ and CAP takes $391s$ for the same number of samples, respectively.

Limitations: There are several limitations of our current approach which can be addressed in future work. 1) Although the sampling framework works well for mesh surfaces, we might fail to compute a valid remeshing if the minimal radius is larger than the local feature size of the surface. For example, we can remesh a thin sheet mesh with parallel planes using small triangles, but we cannot remesh it using large triangles. Another limitation is that we don't have any theoretical guarantee for convergence of the randomized mesh optimization. This is partially due to the fact that in some extreme cases the valence and angle optimization conflict with each other. One possible solution is to further enlarge the region to be resampled. The third limitation is that our framework depends on the regular triangulation/power diagram, which limits the performance of our algorithm compared with GPU based approaches [Bowers et al. 2010].

7 Conclusion and Future Work

We presented a theoretical analysis of gaps in disk sets with varying radii. Based on this analysis, we proposed algorithms and data structures for gap detection and gap updates when the disk set changes. The contribution of our work is illustrated with an adaptive remeshing algorithm that can improve the remeshing quality in aspect of minimal angle, vertex valence and triangle quality. In future work, we would like to build a solution for volumetric remesh-

Model	#v	Q_{min}	θ_{min}	θ_{max}	Hdist	RMS	$\theta_{<30^o}$	v_{567}
adaptive remeshing								
Rockerarm[CVD]	5.8k	0.353	16.5	133.2	0.44	0.048	0.30	98.1
Rockerarm[DSR]	5.8k	0.004	0.14	152.9	0.55	0.057	1.34	93.1
Rockerarm[CVT]	5.8k	0.428	21.3	124.4	**0.34**	**0.032**	0.11	99.7
Rockerarm[CAP]	5.8k	0.413	23.5	126.4	0.58	0.050	0.12	98.8
Rockerarm[MPS]	5.3k	0.320	14.0	130.0	0.48	0.034	1.49	90.6
Rockerarm[OUR]	5.8k	**0.516**	**32.0**	**113.6**	0.48	0.033	**0**	**100**
Homer[CVD]	7.2k	0.049	2.82	173.2	0.35	0.057	2.03	94.3
Homer[DSR]	7.5k	0.150	21.7	129.9	0.43	0.068	0.10	95.2
Homer[CVT]	7.2k	0.334	16.2	120.8	**0.20**	**0.029**	2.04	99.5
Homer[CAP]	7.2k	0.405	21.0	126.6	0.23	0.046	0.42	96.4
Homer[MPS]	7.3k	0.371	21.8	131.4	0.32	0.028	0.29	95.4
Homer[OUR]	7.2k	**0.513**	**32.0**	**115.0**	0.31	**0.023**	**0**	**100**
Triceratops[CVD]	9k	0.007	0.46	179.1	0.59	0.050	4.37	94.1
Triceratops[CVT]	9k	0.385	15.5	127.3	**0.24**	**0.038**	0.29	99.2
Triceratops[CAP]	9k	0.411	21.2	126.4	0.32	0.035	0.43	97.2
Triceratops[MPS]	9k	0.29	13.5	138.6	0.48	0.062	1.23	92.0
Triceratops[OUR]	9k	**0.506**	**32.0**	**114.8**	0.46	0.062	**0**	**100**
Elephant[CVD]	11k	0.040	1.33	173.2	0.38	0.039	4.15	95.4
Elephant[CVT]	11k	0.408	21.2	126.6	**0.23**	**0.024**	0.72	94.0
Elephant[CAP]	11k	0.316	18.5	138.1	0.24	**0.024**	0.04	93.7
Elephant[MPS]	11k	0.301	13.0	130.0	0.46	0.029	1.09	92.6
Elephant[OUR]	11k	**0.505**	**32.0**	**114.9**	0.38	0.061	**0**	**100**
Bunny[CVD]	12k	0.15	9.6	160.1	0.34	**0.028**	0.71	96.0
Bunny[CVT]	12k	0.36	17.8	133.2	**0.20**	0.029	0.38	96.4
Bunny[CAP]	12k	0.20	7.48	137.8	0.48	0.038	4.99	97.5
Bunny[MPS]	9.8k	0.36	19.2	133.2	0.46	0.042	0.90	94.7
Bunny[OUR]	12k	**0.51**	**32.0**	**114.6**	0.37	0.035	**0**	**100**
uniform remeshing								
Joint[CVD]	3.2k	0.040	2.4	174.6	**0.12**	**0.011**	15.6	96.0
Joint[DEL]	3.2k	0.057	2.83	171.5	0.38	0.031	2.6	91.1
Joint[CVT]	3.2k	0.555	29.6	108.4	0.26	0.056	0.005	99.2
Joint[OUR]	3.2k	**0.688**	**33.9**	**104.9**	0.37	0.056	**0**	**99.4**
Bunny[CVD]	12k	0.111	3.81	146.2	0.43	0.037	17.2	97.6
Bunny[CVT]	12k	0.618	37.4	101.2	**0.20**	**0.029**	**0**	**100**
Bunny[CAP]	12k	0.215	20.5	126.0	0.35	**0.029**	0.41	99.5
Bunny[MPS]	12k	0.480	30.3	118.0	0.44	0.037	0	96.5
Bunny[OUR]	12k	**0.629**	**38.0**	**100.0**	0.47	0.034	**0**	**100**
Elk[SAG]	31k	0.092	4.72	166.8	0.30	0.047	0.07	99.7
Elk[CVD]	31k	0.037	2.28	175.0	**0.20**	**0.012**	1.60	96.5
Elk[CVT]	31k	0.634	36.5	99.2	0.21	0.014	**0**	99.9
Elk[CAP]	31k	0.305	11.4	133.4	0.30	0.027	1.93	98.7
Elk[OUR]	31k	**0.645**	**37.0**	**97.9**	0.26	0.020	**0**	**100**

Table 2: *Statistics of the remeshing quality compared with pervious work. #v is the number of sampled points. Q_{min} is the minimal triangle quality [Frey and Borouchaki 1997]; θ_{min} and θ_{max} are the minimal and maximal angle; Hdist and RMS are Hausdorff distance and the root mean square distance between remeshing and the input mesh(% of the diagonal length of the bounding box); v_{567} is the percent of vertices with valences 5,6 and 7. Here SAG refers to [Surazhsky et al. 2003], DEL refers to [Cheng et al. 2007], DSR refers to [Fu and Zhou 2008], CVD refers to [Valette et al. 2008], CVT refers to [Yan et al. 2009], CAP refers to [Chen et al. 2012], MPS refers to our approach without optimization, respectively. The best result is highlighted with bold font.*

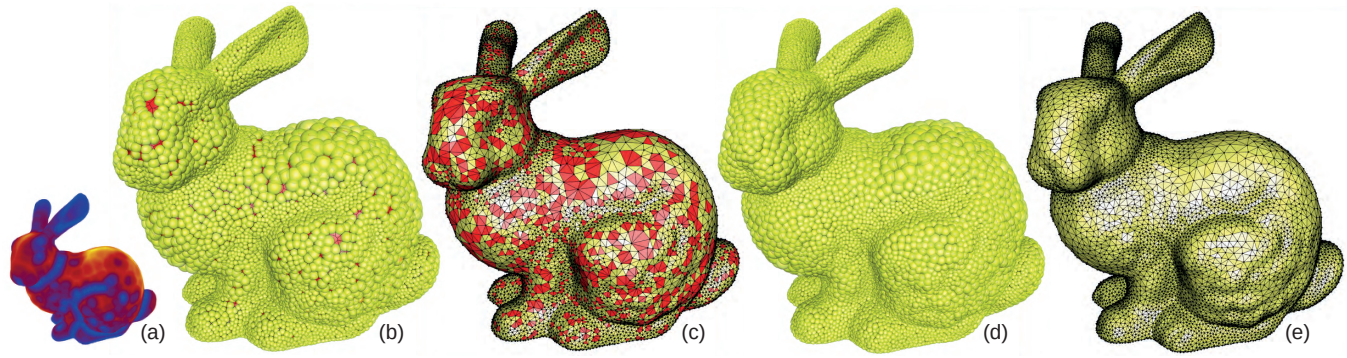


Figure 14: *Adaptive maximal Poisson-disk sampling on the Bunny model. (a) the density map, where cooler colors correspond to a smaller radius while warmer colors correspond to a larger radius; (b) non-maximal sampling results in gaps (red regions); (c) triangles of the remeshing that are effected by gaps (red triangles); (d) maximal sampling; and (e) remeshing using maximal sampling. Only maximal sampling leads to good triangle shapes that are essential for simulation.*

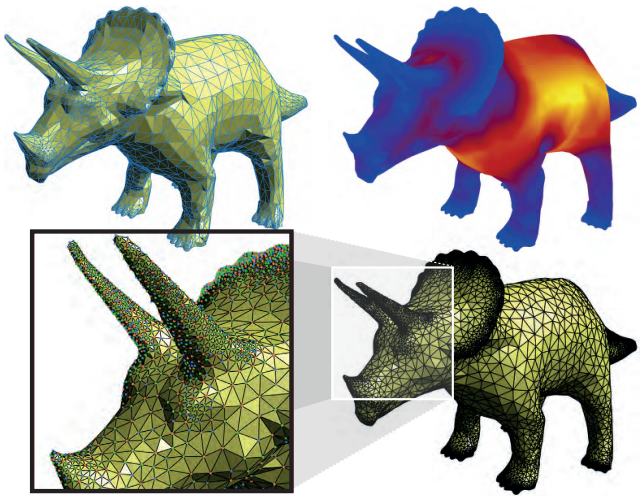


Figure 15: *Adaptive sampling/remeshing. Top: input mesh and the density function; Bottom: remeshing result.*

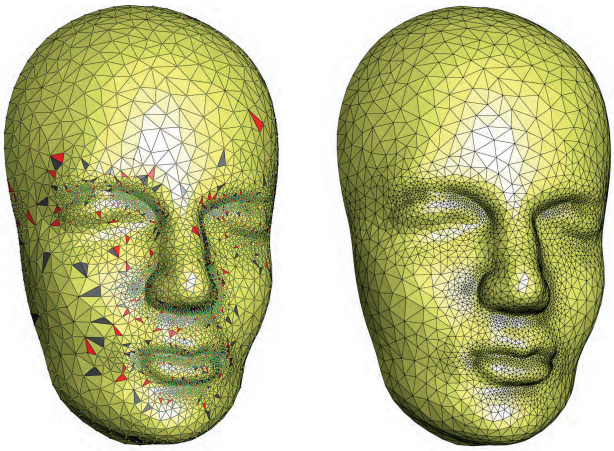


Figure 16: *Boundary handling, 9k vertices. (a) maximal sampling/remeshing. The triangles with $\theta_{min} < 30^o$ are show in dark gray and the triangles with $\theta_{min} \in [30^o, 32^o]$ are shown in red. (b) Optimized sampling and remeshing. The optimized angle range is $[32^o, 115^o]$.*

ing that can be applied to static and deformable objects.

References

Alliez, P., Ucelli, G., Gotsman, C., and Attene, M. 2008. Recent advances in remeshing of surfaces. In *Shape Analysis and Structuring*, 53–82.

Amenta, N., and Bern, M. 1999. Surface reconstruction by Voronoi filtering. *Discrete and Computational Geometry 22*, 481–504.

Amenta, N., Bern, M., and Kamvysselis, M. 1998. A new Voronoi-based surface reconstruction algorithm. In *Proc. ACM SIGGRAPH*, 415–421.

Ash, P. F., and Bolker, E. D. 1986. Generalized Dirichlet tessellations. *Geometriae Dedicata 20*, 2, 209243.

Aurenhammer, F. 1991. Voronoi diagrams—A survey of a fundamental geometric data structure. *ACM Comput. Surveys 23*, 3, 345405.

Balzer, M., Schlömer, T., and Deussen, O. 2009. Capacity-constrained point distributions: A variant of Lloyds method. *ACM Trans. on Graphics (Proc. SIGGRAPH) 28*, 6, 86:1–86:8.

Bowers, J., Wang, R., Wei, L.-Y., and Maletz, D. 2010. Parallel Poisson disk sampling with spectrum analysis on surfaces. *ACM Trans. on Graphics (Proc. SIGGRAPH Asia) 29*, 6, 166:1–166:10.

Bridson, R. 2007. Fast Poisson disk sampling in arbitrary dimensions. In *ACM SIGGRAPH 2007 Sketches*.

CGAL, Computational Geometry Algorithms Library. http://www.cgal.org.

Chen, Z., Yuan, Z., Choi, Y.-K., Liu, L., and Wang, W. 2012. Variational blue noise sampling. *IEEE Trans. on Vis. and Comp. Graphics 18*, 10, 1784–1796.

Cheng, S.-W., Dey, T. K., and Levine, J. A. 2007. A practical Delaunay meshing algorithm for a large class of domains. In *16th Intl. Meshing Roundtable*, 477–494.

Chew, L. P. 1989. Guaranteed-quality triangular meshes. Department of computer science tech report 89-983, Cornell University.

Cline, D., Jeschke, S., Razdan, A., White, K., and Wonka, P. 2009. Dart throwing on surfaces. *Computer Graphics Forum (Proc. EGSR) 28*, 4, 1217–1226.

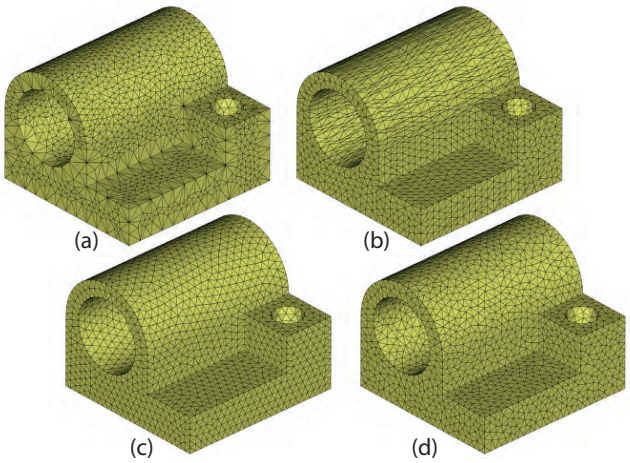


Figure 17: *Comparison of the remeshed Joint model, 3.2k vertices. (a) DelPSC [Cheng et al. 2007]; (b) CVD [Valette et al. 2008]; (c) CVT [Yan et al. 2009]; (d) ours.*

Cook, R. L. 1986. Stochastic sampling in computer graphics. *ACM Trans. on Graphics 5*, 1, 69–78.

Corsini, M., Cignoni, P., and Scopigno, R. 2012. Efficient and flexible sampling with blue noise properties of triangular meshes. *IEEE Trans. on Vis. and Comp. Graphics 18*, 6, 914–924.

de Castro, P. M. M., Tournois, J., Alliez, P., and Devillers, O. 2009. Filtering relocations on a Delaunay triangulation. *Computer Graphics Forum (Proc. SGP) 28*, 5, 1465–1474.

Dippé, M. A. Z., and Wold, E. H. 1985. Antialiasing through stochastic sampling. *Proc. ACM SIGGRAPH*, 69–78.

Dunbar, D., and Humphreys, G. 2006. A spatial data structure for fast Poisson-disk sample generation. *ACM Trans. on Graphics (Proc. SIGGRAPH) 25*, 3, 503–508.

Ebeida, M. S., Mitchell, S. A., Davidson, A. A., Patney, A., Knupp, P. M., and Owens, J. D. 2011. Efficient and good Delaunay meshes from random points. *Computer-Aided Design 43*, 11, 1506–1515.

Ebeida, M. S., Patney, A., Mitchell, S. A., Andrew Davidson, P. M. K., and Owens, J. D. 2011. Efficient maximal Poisson-disk sampling. *ACM Trans. on Graphics (Proc. SIGGRAPH) 30*, 4, 49:1–49:12.

Ebeida, M. S., Mitchell, S. A., Patney, A., Davidson, A. A., and Owens, J. D. 2012. A simple algorithm for maximal Poisson-disk sampling in high dimensions. *Computer Graphics Forum (Proc. EUROGRAPHICS) 31*, 2, 785–794.

Edelsbrunner, H., and Shah, N. R. 1997. Triangulating topological spaces. *IJCGA 7*, 4, 365–378.

Fattal, R. 2011. Blue-noise point sampling using kernel density model. *ACM Trans. on Graphics (Proc. SIGGRAPH) 28*, 3, 48:1–48:10.

Frey, P., and Borouchaki, H. 1997. Surface mesh evaluation. In *6th Intl. Meshing Roundtable*, 363–374.

Fu, Y., and Zhou, B. 2008. Direct sampling on surfaces for high quality remeshing. In *ACM symposium on Solid and physical modeling*, 115–124.

Gamito, M. N., and Maddock, S. C. 2009. Accurate multidimensional Poisson-disk sampling. *ACM Trans. on Graphics 29*, 1, 8:1–8:19.

Jones, T. R. 2006. Efficient generation of Poisson-disk sampling patterns. *Journal of Graphics Tools 11*, 2, 27–36.

Kalantari, N. K., and Sen, P. 2011. Efficient computation of blue noise point sets through importance sampling. *Computer Graphics Forum (Proc. EGSR) 30*, 4, 1215–1221.

Kopf, J., Cohen-Or, D., Deussen, O., and Lischinski, D. 2006. Recursive Wang tiles for real-time blue noise. *ACM Trans. on Graphics (Proc. SIGGRAPH) 11*, 2, 509–518.

Lagae, A., and Dutré, P. 2008. A comparison of methods for generating Poisson disk distributions. *Computer Graphics Forum 27*, 1, 114–129.

Lloyd, S. 1982. Least squares quantization in PCM. *IEEE Transactions on Information Theory 28*, 2, 129–137.

McCool, M., and Fiume, E. 1992. Hierarchical Poisson disk sampling distributions. In *Graphics Interface*, 94–105.

Mitchell, S. A., Rand, A., Ebeida, M. S., and Bajaj, C. L. 2012. Variable radii poisson disk sampling. 185–190.

Mitchell, D. P. 1987. Generating antialiased images at low sampling densities. In *Proc. ACM SIGGRAPH*, 65–72.

Mitchell, D. P. 1991. Spectrally optimal sampling for distribution ray tracing. In *Proc. ACM SIGGRAPH*, 157–164.

Nivoliers, V., Yan, D.-M., and Lévy, B. 2011. Fitting polynomial surfaces to triangular meshes with Voronoi squared distance minimization. In *International Meshing Roundtable conference proceedings*, 601–618.

Ostromoukhov, V., Donohue, C., and Jodoin, P.-M. 2004. Fast hierarchical importance sampling with blue noise properties. *ACM Trans. on Graphics (Proc. SIGGRAPH) 23*, 3, 488–495.

Shewchuk, J. R. 2002. What is a good linear element? interpolation, conditioning, and quality measures. In *11th Intl. Meshing Roundtable*, 115–126.

Surazhsky, V., Alliez, P., and Gotsman, C. 2003. Isotropic remeshing of surfaces: a local parameterization approach. In *12th Intl. Meshing Roundtable*, 204–231.

Turk, G. 1993. Generating random points in triangles. In *Graphics Gems*, 24–28.

Valette, S., Chassery, J.-M., and Prost, R. 2008. Generic remeshing of 3D triangular meshes with metric-dependent discrete Voronoi diagrams. *IEEE Trans. on Vis. and Comp. Graphics 14*, 2, 369–381.

Wei, L.-Y., and Wang, R. 2011. Differential domain analysis for non-uniform sampling. *ACM Trans. on Graphics (Proc. SIGGRAPH) 30*, 4, 50:1–50:8.

Wei, L.-Y. 2008. Parallel Poisson disk sampling. *ACM Trans. on Graphics (Proc. SIGGRAPH) 27*, 3, 20:1–20:9.

Wei, L.-Y. 2010. Multi-class blue noise sampling. *ACM Trans. on Graphics (Proc. SIGGRAPH) 29*, 4, 79:1–79:8.

White, K. B., Cline, D., and Egbert, P. K. 2007. Poisson disk point sets by hierarchical dart throwing. In *Proceedings of the IEEE Symposium on Interactive Ray Tracing*, 129–132.

Xu, Y., Hu, R., Gotsman, C., and Liu, L. 2012. Blue noise sampling of surfaces. *Computers & Graphics 36*, 4, 232–240.

Yan, D.-M., Lévy, B., Liu, Y., Sun, F., and Wang, W. 2009. Isotropic remeshing with fast and exact computation of restricted Voronoi diagram. *Computer Graphics Forum 28*, 5, 1445–1454.

Yan, D.-M., Wang, W., Lévy, B., and Liu, Y. 2012. Efficient computation of clipped Voronoi diagram for mesh generation. *Computer-Aided Design xx*, x, to appear.

A Gap decomposition

In this appendix, we first discuss some properties of gaps and gap triangles, then we show that the algorithm presented in Sec. 3.3 gives a valid decomposition.

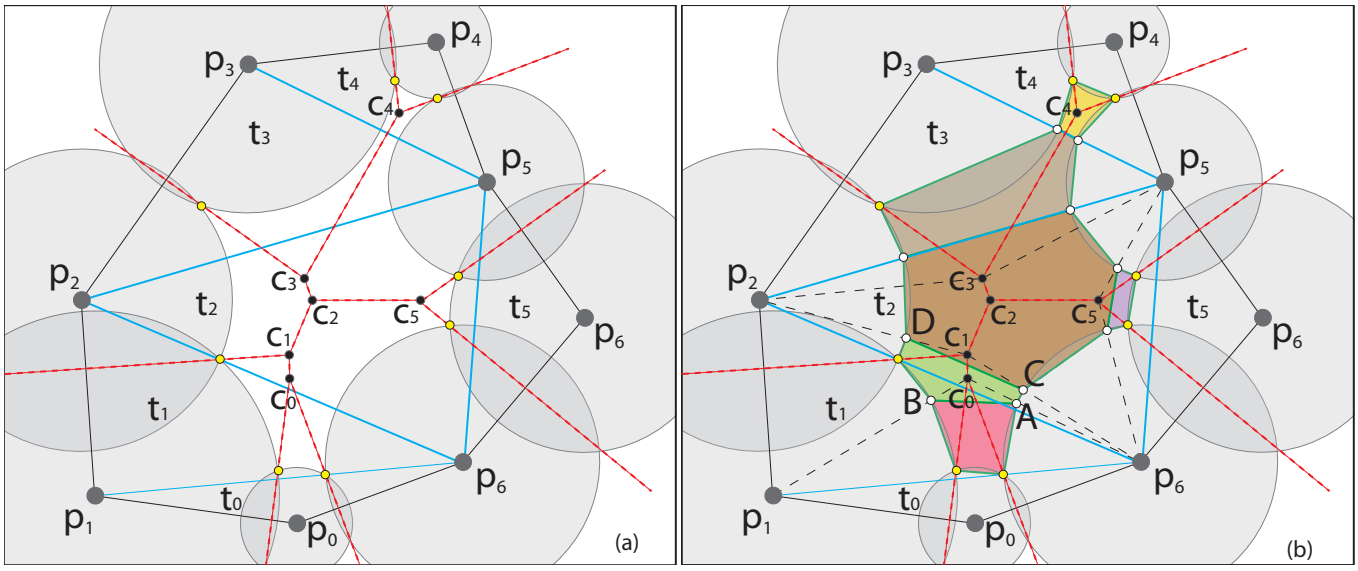


Figure 18: *An illustration of the case that multiple power centers fall into one triangle. Left: power diagram (red) and regular triangulation (solid black and blue). Five power centers fall into the same triangle t_2. Right: our algorithm handles this case correctly. The gap is decomposed into five polygons w.r.t. five gap triangles.*

Recall that a connected gap component is incident to a set of gap triangles which cover the gap. The gap triangles of a gap can be clustered together using the connectivity rules (Sec. 3.2) without applying the last rule (Fig. 4(d)).

There are two types of edges of gap triangles as shown in Fig 18: boundary edges (solid black) and inner edges (blue). A boundary edge has edge length smaller than the sum of the radii of its two endpoints, i.e., $|\mathbf{p}_i\,\mathbf{p}_j| < r_i + r_j$. The boundary edges isolate the gap from other gaps. The inner edges have two neighboring triangles that are incident to the same gap. The gap primitive of each gap triangle is extracted by traversing the edges of the triangle in ccw order. Each boundary edge contributes one point to the gap primitive. The point is the intersection point of two circles centered at the vertices of the boundary edge (yellow dots in Fig. 18(a)). Each inner edge splits the gap into two components, so that an inner edge (blue) contributes a segment to the gap primitive of two neighboring triangles.

The inner edges can be further classified into two types: (1) inner edges with two power centers of neighboring gap triangles that lie on different sides of the edge, e.g., edge $\mathbf{p}_3\,\mathbf{p}_5$ in Fig. 18(a); (2) inner edges with two power centers of neighboring gap triangles that lie on the same sides of the edge, e.g., edges $\mathbf{p}_1\,\mathbf{p}_6$, $\mathbf{p}_2\,\mathbf{p}_6$,$\mathbf{p}_2\,\mathbf{p}_5$,$\mathbf{p}_3\,\mathbf{p}_5$ and $\mathbf{p}_5\,\mathbf{p}_6$. For the first type of edge, we simply compute the intersection points of the edge and two circles centered at the endpoints of the edge and connect the intersection points, which split the gap into two components. For each second type inner edge, we compute an auxiliary segment, which also splits the gap into two components. Intersection points are shown in white (see Fig. 18(b)).

The only problem that remains is to show that the splitting edges do not intersect with each other. To see this, we only need to show that two splitting edges of the same triangle do not intersect. We use gap triangle t_1 for example. t_1 has two inner edges, i.e., $\mathbf{p}_1\mathbf{p}_6$ and $\mathbf{p}_2\mathbf{p}_6$. By applying the algorithm presented in Sec. 3.3, we compute two splitting edges, AB and CD w.r.t. $\mathbf{p}_1\mathbf{p}_6$ and $\mathbf{p}_2\mathbf{p}_6$. The intersection points A and C are from the same circle centered at $\mathbf{p}_6$, and points B and D are from circles centered at $\mathbf{p}_1$ and $\mathbf{p}_2$, respectively. Since quads ACc_1c_0 and c_0c_1DB are in ccw order, AB and CD cannot intersect. We can conclude that none of splitting edges intersect with each other and each of them splits the gap into two regions.

To this end, we conclude that our decomposition algorithm fulfills the following requirements:

- Each gap triangle is associated with a simple polygon with up to six edges, since each edge of a gap triangle contributes at most two vertices to the gap primitive of a triangle.
- All these polygons are non-intersecting. Since the edges of a polygon are either a chord of the circle, or the bi-sectors of the gap, so that none of gap primitives intersect except along the common edges.
- The union of these polygons covers the whole gap. To see this, just note that each polygon covers its associated sub-gap since all the vertices of the polygon are on the circles and the sub-gaps are the area bounded by these circles.